我们这个课程的系统是怎么搭建起来的？

2021-05-28 高楼

《高楼的性能工程实战课》 课程介绍 >

讲述：高楼

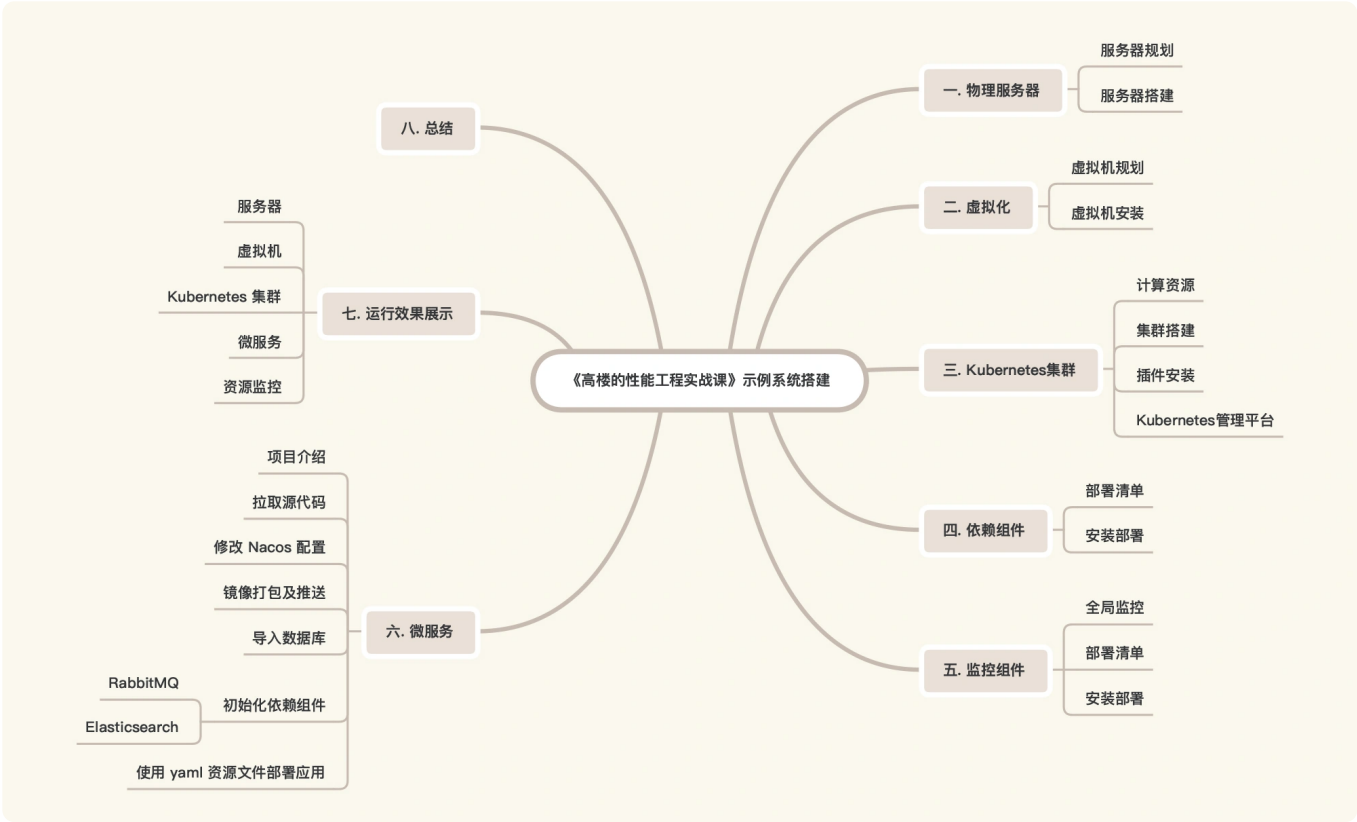
时长 03:40 大小 3.38M

你好，我是高楼。

在我们这个课程里，为了让你更好地理解我的性能工程理念，我专门搭建了一个完整的系统，我们所有的内容都是基于这个系统展开的。

自课程更新以来，有不少同学问我要这个系统的搭建教程，想自己试一试。因此，我梳理了一版搭建教程，希望能帮到你。

由于整个系统相对复杂，有很多需要考虑、部署的细节，所以这节课的内容会比较长。下面这张图是我们这节课的目录，你可以整体了解一下，然后对应这张目录图，来学习具体的搭建步骤，以免迷失方向。



一. 物理服务器

1. 服务器规划

在这个系统中，我们主要用到了四台服务器，下面是具体的硬件配置：

序号	主机	配置	用途	备注
1	物理服务器 1	KVM 服务器	16C-32G-280G	root@dell-server-2 ~# free -h total used free shared buff/cache available Mem: 31G 29G 222M 8.4M 1.0G 711M Swap: 63G 13G 50G [root@dell-server-2 ~]#
2	物理服务器 2	KVM 服务器	16C-32G-280G	root@dell-server-3 ~# free -h total used free shared buff/cache available Mem: 31G 30G 190M 4.2M 437M 163M Swap: 63G 26G 37G [root@dell-server-3 ~]#
3	物理服务器 3	KVM 服务器	32C-64G-1T	root@hp-server ~# free -h total used free shared buff/cache available Mem: 62G 52G 1.1G 61M 9.4G 8.5G Swap: 0B 0B 0B [root@hp-server ~]#
4	物理服务器 4	NFS 服务器	4C-4G-300G	root@lenvo-nfs-server ~# free -h total used free shared buff/cache available Mem: 3.4G 917M 136M 165M 2.4G 2.0G Swap: 8.0G 110M 7.9G [root@lenvo-nfs-server ~]#

我们可以看到，当前服务器在应用中使用的资源总共是 64C 的 CPU 资源，以及 128 G 的内存资源。由于 NFS (网络存储) 服务器不用在应用中，我们不计算在内。

因为单台机器的硬件资源相对较多，所以，在后续的工作中，我们将这些物理机化为虚拟机使用，以方便应用的管理。

在成本上，所有物理机的费用加在一起大概八万元左右，这其中还包括交换机、机柜、网线等各类杂七杂八的费用。

2. 服务器搭建

目前，行业内主流的基于 x86 架构的 Linux 系统，无非是 CentOS 和 Ubuntu。在我们这个项目中，我选择 CentOS 系列来搭建 Linux 系统，主要是考虑到系统的稳定性。CentOS 来自 Redhat 商业版本的重新编译，它在稳定性、系统优化以及兼容性方面，具有比较完善的测试和发版流程。

在 CentOS 7 之后的版本中，CentOS 的内核换成了 Linux 3.x，因此，我们这个课程的分析都是基于 Linux 3.x 这个内核版本展开的。

在搭建过程中，我们给每台服务器都安装了 CentOS 7.8 的操作系统。如果你是新手，我建议你使用带 GUI 桌面的系统，方便后续操作和管理虚拟机。具体的操作系统安装步骤，你可以参考这个链接来部署： HP 服务器安装 CentOS 7 。

二. 虚拟化

1. 虚拟机规划

我们接着来看虚拟机规划。我们部署了至少五台虚机，并且把虚拟机类型分为两种主机节点类型：

普通节点：

普通节点用来做非被测系统使用，比如压力机、管理平台等。我们可以选择采用 Docker、二进制等方式来部署。

Kubernetes 节点：

Kubernetes 节点用于部署项目的应用服务，包括 mall-admin、mall-portal、mall-gateway、mall-member、mall-cart 等应用服务，还包括 SkyWalking、Nacos 等基础组件。这些都采用 Kubernetes 的方式来部署。

具体的节点规划，你可以参考这张表：

主机	操作系统	最低配置	说明
master-1	CentOS 7.8	2C-4G-50G	Kubernetes 控制节点
master-2	CentOS 7.8	2C-4G-50G	Kubernetes 控制节点
master-3	CentOS 7.8	2C-4G-50G	Kubernetes 控制节点
worker-1	CentOS 7.8	4C-8G-50G	Kubernetes 计算节点
worker-2	CentOS 7.8	4C-8G-50G	Kubernetes计算节点
worker-3	CentOS 7.8	4C-8G-50G	Kubernetes 计算节点
.....	CentOS 7.8	4C-8G-50G	Kubernetes 计算节点
nfs-server	CentOS 7.8	2C-4G-500G	网络存储节点
linux-server-1	CentOS 7.8	4C-8G-50G	普通节点
.....	CentOS 7.8	4C-8G-50G	普通节点

在这里，我们规划了三个 Kubernetes 控制节点，这是为后续的高可用方案准备的。如果你计划搭建单 Master 集群，只需要一个 Kubernetes 控制节点即可。至于 Kubernetes 计算节点，结合前面的节点规划，我们在这里配置 9 个 worker 节点，其他的节点根据自己的需求灵活扩展。

2. 虚机安装

到了安装虚拟机这一步，我们最终选择以 KVM 为主的方案。这主要考虑到，KVM 是目前比较成熟的开源虚拟化平台，在 2006 年被写入到 Linux 内核中。并且在 RedHat 6 以后，RedHat 开始转向支持 KVM，而非之前大力推广的 Xen 虚拟化方案，随后 Intel 也开始全面支持 KVM。KVM 相比较于 Xen，更小，更轻量级，也更方便管理。

在项目搭建之初，我们也尝试过用 OpenStack 做底层，但是 OpenStack 部署起来不仅繁杂，而且坑也多，需要投入大量的时间成本。我们当时在分析 OpenStack 本身的问题上花费了很多时间，对于我们的这个系统来说，这是没有必要的。

所以，我们最终选择用 KVM 来做虚拟化，它的技术相对成熟，操作又比较简单。

你可能会有疑问，为什么不用 VMware 呢？我们知道，在虚拟化平台中，VMware 在 IO 和稳定性方面都算是目前最优的一个方案了，也能满足我们的需求。不过，它是一款商业软件，授权比较昂贵，这是我们这个项目不得不放弃的一个原因。当然，如果你的项目有充足的预算， VMware 是一个不错的选择。

在安装之前，你可以大概了解一下 KVM 性能、热迁移、稳定性、应用移植、搭建等方面的注意事项，做为知识的扩展补充。对性能分析来说，我们要关注一下 KVM 的优化重点：关于 KVM 虚拟化注意的二三事整理

至于 KVM 的安装和使用，你可以参考这个链接里的内容：Linux KVM 安装使用手册。

三. Kubernetes 集群

1. 计算资源

关于集群计算资源，你可以参考这张表：

	物理机	云主机	自行准备的虚机
I/O 损耗	低	稍微	高
成本	中	低	高
额外资源开销	低	中	高
灵活度	高	低	低

我们在做计算资源规划的时候，通常需要考虑不同的应用场景：

传统虚拟化技术的 I/O 损耗较大，对于 I/O 密集型应用，物理机相比传统虚拟机（像 VMware 的传统虚拟化做出来的虚拟机) 有更好的性能表现；

在物理机上部署应用，有更少的额外资源开销（如虚拟化管理、虚拟机操作系统等），并且可以有更高的部署密度，来降低基础设施成本；

在物理机上可以更加灵活地选择网络、存储等设备和软件应用生态。

如果从实际生产环境考虑，一般而言建议：

对性能极其敏感的应用，如高性能计算，物理机是较好的选择；

云主机支持热迁移，可以有效降低运维成本；

在工作实践中，我们会为 Kubernetes 集群划分静态资源池和弹性资源池。通常而言，固定资源池可以根据需要选择物理机或者云主机实例；弹性资源池则可以根据应用负载，使用合适规格的云主机实例来优化成本，避免资源浪费，同时提升弹性供给保障。

由于我们这个系统只是课程的示例项目，为了尽可能压榨服务器资源，节省服务器成本，我们选择了自行准备虚机的方案，这样可以充分使用硬件资源。

2. 集群搭建

关于集群搭建，我们的节点规划如下：

序号	主机	IP地址	Kubernetes版本	安装组件	备注
1	K8s-master-1	172.16.106.208	v1.19.2	calico v3.13.1 apisever v1.19.2 etcd v3.4.13 scheduler v1.19.2 controller-manager v1.19.2 proxy v1.19.2	master leader
2	K8s-master-2	172.16.106.80			master follow
3	K8s-master-3	172.16.106.14			master follow
4	K8s-worker-1	172.16.106.169		calico v3.13.1 proxy v1.19.2	worker nodes
5	K8s-worker-2	172.16.106.149			
6	K8s-worker-3	172.16.106.195			
7	K8s-worker-4	172.16.106.55			
8	K8s-worker-5	172.16.106.100			
9	K8s-worker-6	172.16.106.79			
10	K8s-worker-7	172.16.106.150			
11	K8s-worker-8	172.16.106.56			
12	K8s-worker-9	172.16.106.129			

关于集群搭建的具体步骤，你可以按照下面这两个文档进行部署：

单 Master 集群：使用 kubeadm 安装单 master kubernetes 集群（脚本版）

高可用方案： Kubernetes 高可用集群落地二三事

安装的负载均衡组件如下：

组件	版本号	说明
Nginx Ingress	2.2.8	Ingress 能够集成授权入站，并连接到集群服务的规则集合。此外，Ingress 提供七层负载均衡能力，可配置提供外部可访问的 URL、负载均衡、SSL、基于名称的虚拟主机等功能。作为集群流量接入层，Ingress 具有高可靠性。

如果你没有 Kubernetes 的使用基础，那么我建议学习一下这几篇入门文章：

Kubernetes 集群基本概念

k8s 入门篇 -Kubernetes 的基本概念和术语

K8s 命令篇 -Kubernetes 工作实用命令集结号

Kubernetes 集群常用操作总结

3. 插件安装

我们需要安装的插件主要有三种：网络插件、存储插件以及组件。

对于网络插件，我们选用的是目前主流的网络插件 Calico。如果你的系统有其它选型需求，那你可以参考下面这篇文章，这里我就不做赘述了。

Kubernetes 网络插件（CNI）超过 10Gbit/s 的基准测试结果

安装 Calico 插件的具体步骤，在前面的单 Master 集群部署文档中已有说明，你可以参考一下。

对于存储插件，我们选用的是 NFS 网络存储。因为 NFS 相对简单，上手快，我们只需要部署一个 NFS 服务，再由 Kubernetes 提供一个自动配置卷程序，然后通过 StoageClass 动态配置 PVC 就可以了。 而且在性能上，NFS 也能满足我们这个系统的需求。

只不过，NFS 并不是高可用方案。如果你是在生产环境中使用，可以考虑把 Ceph 作为存储选型方案。Ceph 是一个统一的分布式存储系统，也是高可用存储方案，并且可以提供比较好的性能、可靠性和可扩展性。但是，Ceph 部署起来更复杂些，同时维护也比 NFS 复杂。

我把 NFS 和 Ceph 的详细安装步骤放在这里，你如果有需要，可以学习参考。

NFS： Kubernetes 集群部署 NFS 网络存储

Ceph： Kubernetes 集群分布式存储插件 Rook Ceph 部署

另外，不要忘了，NFS 配置中还需要这两个组件：

组件	版本号	说明
nfs-utils	1.3.0	NFS 客户端
nfs-client-provisioner	3.1.0	NFS 自动配置卷程序

4. Kubernetes 管理平台

安装组件：

组件	版本号	说明
Kuboard	2.0.5.5	微服务管理工具

Kuboard 采用的是可视化 UI 的方式来管理应用和组件，降低了 Kubernetes 集群的使用门槛。下面我们看看怎么部署 Kuboard 组件。

第一步，k8s 集群执行资源文件：

```
1 kubectl apply -f https://kuboard.cn/install-script/kuboard.yaml
2 kubectl apply -f https://addons.kuboard.cn/metrics-server/0.3.7/metrics-server
```

第二步，把 Kuboard 安装好后，我们看一下 Kuboard 的运行状态：

```
1 kubectl get pods -l k8s.kuboard.cn/name=kuboard -n kube-system
```

输出结果：

```
NAME                          READY   STATUS        RESTARTS   AGE
kuboard-54c9c4f6cb-6lf88      1/1     Running       0          45s
```

复制代码

这个结果表明 kuboard 已经成功部署了。

接着，我们获取管理员 Token 。这一步是为了登录访问 Kuboard，检查组件是否成功运行。

```
# 可在第一个 Master 节点上执行此命令
echo $(kubectl -n kube-system get secret $(kubectl -n kube-system get secret |
```

复制代码

通过检查部署我们了解到，Kuboard Service 使用了 NodePort 的方式暴露服务，NodePort 为 32567。因此，我们可以按照下面这个方式访问 Kuboard：

```
http://任意一个Worker节点的IP地址:32567/
```

复制代码

然后，在登录中输入管理员 Token，就可以进入到 Kuboard 集群的概览页了。

注意，如果你使用的是阿里云、腾讯云等云服务，那么你可以在对应的安全组设置里，开放 worker 节点 32567 端口的入站访问，你也可以修改 Kuboard.yaml 文件，使用自己定义的 NodePort 端口号。

四. 依赖组件

1. 部署清单

组件	版本号	说明
MySQL	5.7	关系型数据库
Redis	5.0	分布式缓存
Elasticsearch	7.6.2	搜索引擎
Kibana	7.6.2	日志可视化看板
MongoDB	4.2.5	文档数据库
RabbitMQ	3.7.14	消息队列
Helm 3	3.2.4	Kubernetes 包管理器
JMeter	5.4	压测引擎
Nacos	1.4.0	服务注册及配置中心
Logstash	7.6.2	应用日志收集
Harbor	2.0.5.5	镜像仓库
Sentinel	1.8.0	系统保护

2. 安装部署

对于上述依赖组件的安装部署，我整理了对应的教程放在这里，你有兴趣可以尝试一下。

MySQL 的二进制安装方式，在网上的教程多如牛毛，我在这里就不介绍了，如果你想知道怎么在 Kubernetes 下部署 MySQL，你可以参考这个链接中的详细步骤：如何在 Kubernetes 集群中搭建一个复杂的 MySQL 数据库。

Elasticsearch 集群的部署可以参考：

Kubernetes Helm3 部署 Elastic s earch & Kibana 7 集群

JMeter 的部署可以参考：

二进制：性能工具之 JMeter+InfluxDB+Grafana 打造压测可视化实时监控

Kubernetes：Kubernetes 下部署 Jmeter 集群

镜像仓库 Harbor 的部署可以参考：

Kubernetes 集群仓库 harbor Helm3 部署

Nacos 的部署可以参考：

Docker 单机模式： Nacos Docker 快速开始

Kubernetes： Kubernetes Nacos

Redis、RabbitMQ、MongoDB 单机部署的部署可以参考：

Kubernetes 集群监控 kube-prometheus 自动发现

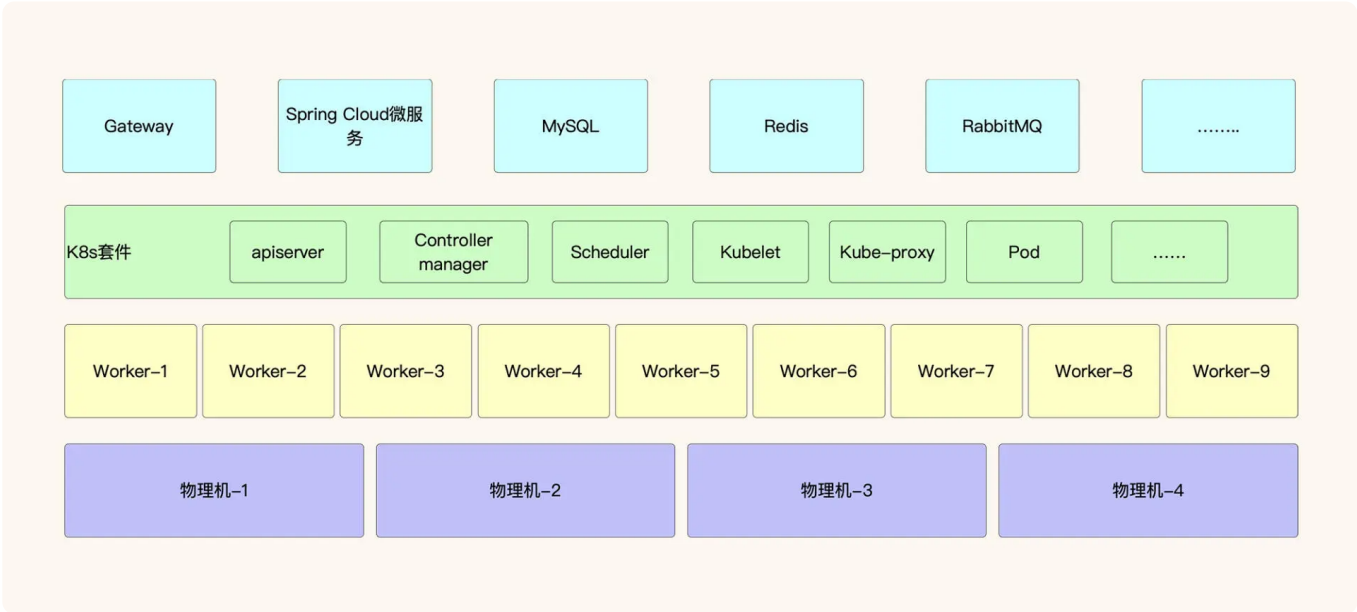
Logstash 的部署可以参考：

整合 ELK 实现日志收集

五. 监控组件

1. 全局监控

不知道你还记不记得，我们这个系统的架构：



根据这个系统的架构，我们选择的工具要监控到这几个层面：

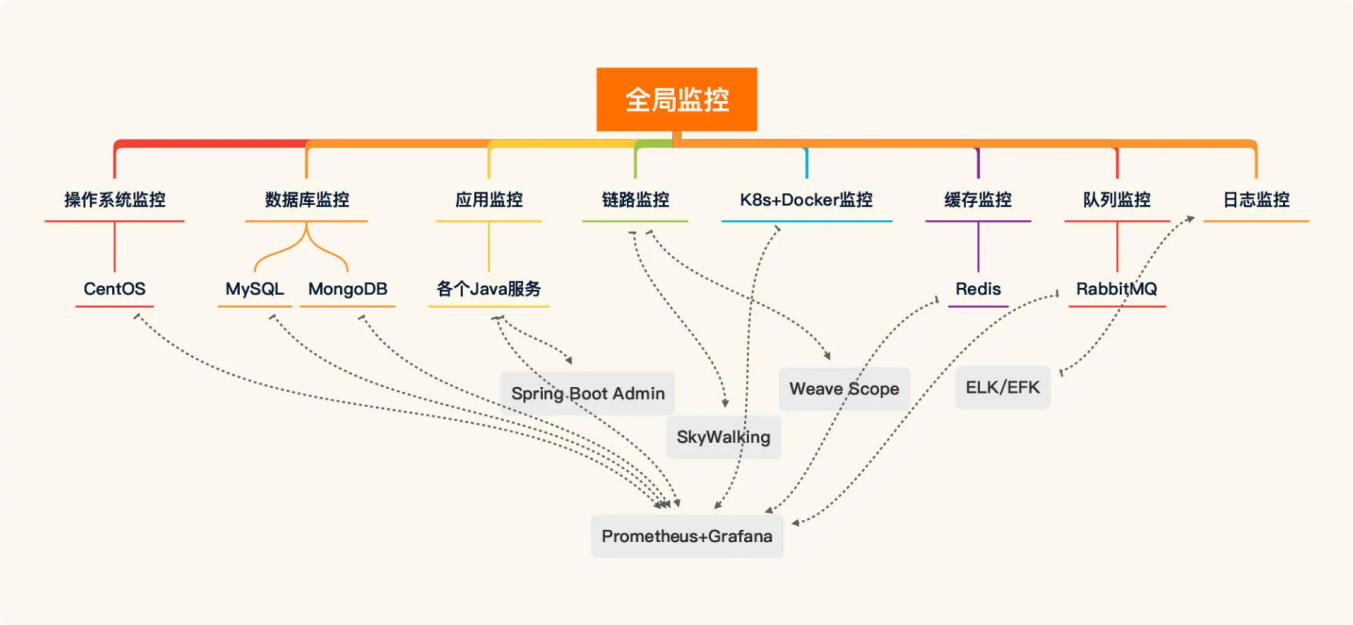
第一层，物理主机；

第二层，KVM 虚拟机；

第三层，Kubernetes 套件；

第四层，各种应用所需要的技术组件。

其实，有了上面的系统架构，监控设计就已经出现在写方案之人的脑袋里了。对于我们这个课程所用的系统，全局监控如下所示：



从上图来看，我们使用 Prometheus/Grafana/Spring Boot Admin/SkyWalking/Weave Scope/ELK/EFK 就可以实现具有全局视角的第一层监控。对于工具中没有覆盖的第一层计数器，我们只能在执行场景时再执行命令来补充了。

2. 部署清单

组件	版本号	说明
SkyWalking	8.1.0	分布式跟踪系统
Prometheus Operator	0.6.0	资源监控系统
Weave Scope	1.13.2	Kubernetes 可视化监控工具
Fluentd	3.0.2	容器日志收集
redis_exporter	1.23.0	Redis 资源监控
mysqld_exporter	0.12.1	MySQL 数据库资源监控
mongodb_exporter	0.20.4	MongoDB 资源监控
rabbitmq_exporter	1.0.0	消息队列资源监控

3. 安装部署

对于上面这些监控工具的部署，我也把相应的安装教程放在这里，供你参考学习。

Kubernetes 集群资源监控的部署：

- Kubernetes 集群监控 kube-prometheus 部署
- Kubernetes 集群监控 controller-manager & scheduler 组件
- Kubernetes 集群监控 ETCD 组件

日志聚合部署的部署：

- Kubernetes 集群日志监控 EFK 安装

依赖组件的部署：

- Kubernetes 集群监控 kube-prometheus 自动发现

APM 链路跟踪的部署：

Kubernetes + Spring Cloud 集成链路追踪 SkyWalking

六. 微服务

1. 项目介绍

在搭建这个课程所用的系统时，我采用了微服务的架构，这也是当前主流的技术架构。

如果你有兴趣了解详细的项目介绍，可以参考这篇文章：《高楼的性能工程实战课》微服务电商项目技术全解析。这里面主要介绍了该项目的一些预备知识、系统结构、主要技术栈以及核心组件。此外，还有相关的运行效果截图。

2. 拉取源代码

我们把 git clone 项目源代码下载到本地，来部署我们的被测系统：

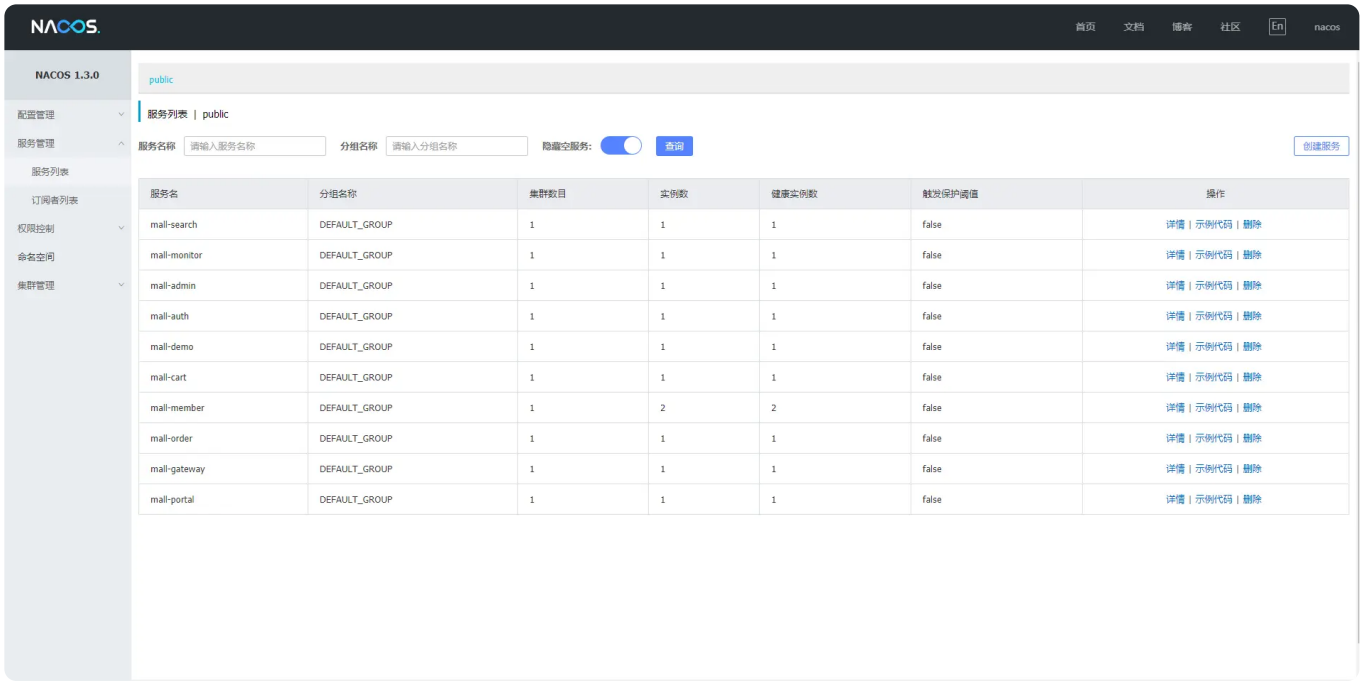
```
1 git clone https://github.com/xncssj/7d-mall-microservice.git
```

复制代码

3. 修改 Nacos 配置

我们先将项目 config 目录下的配置包导入到 Nacos 中，然后根据自己的实际需要修改相关配置。

接着，我们将配置信息导入到 Nacos 中后，会显示这样的信息：



请你注意，我们修改的配置文件主要是每个单体服务下的 application-prod.yml 和 bootstrap-prod.yml。因为两个全局配置文件，都是服务容器内加载的配置文件。

4. 镜像打包及推送

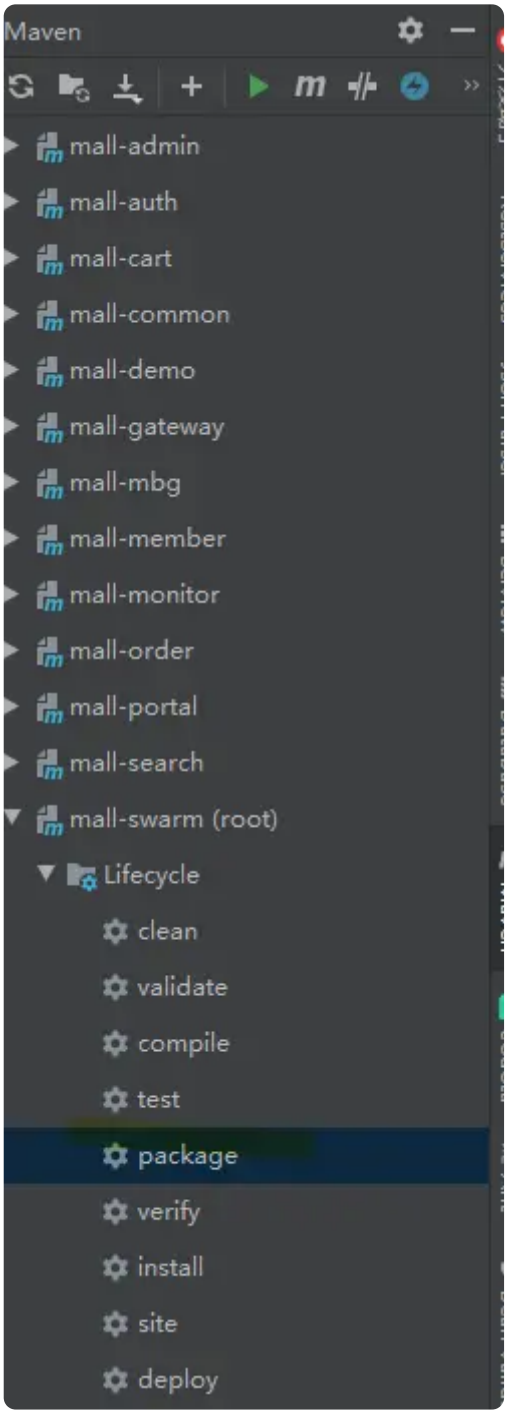
我们使用 Java 语言的 IDE （推荐 IDEA ）打开项目工程。

首先，修改项目根目录下的 pom.xml 文件：

复制代码

```
<properties>
    <!--改为你自己的 Docker 服务远程访问地址-->
    <docker.host>http://172.16.106.237:2375</docker.host>
</properties>
```

在 IDEA 的右边 Maven 标签页，我们可以找到 root 工程下的 package 按钮，选中并执行：



然后，在编译的远程 Docker 主机上，我们修改所有服务的镜像标签名称。之后，再推送镜像到 Docker 仓库。

5. 导入数据库

这一步需要将项目 document/sql 目录下的 SQL 脚本导入到 MySQL 数据库中。

6. 初始化依赖组件

6.1. RabbitMQ

第一步，进入 RabbitMQ 容器并开启管理功能：

复制代码

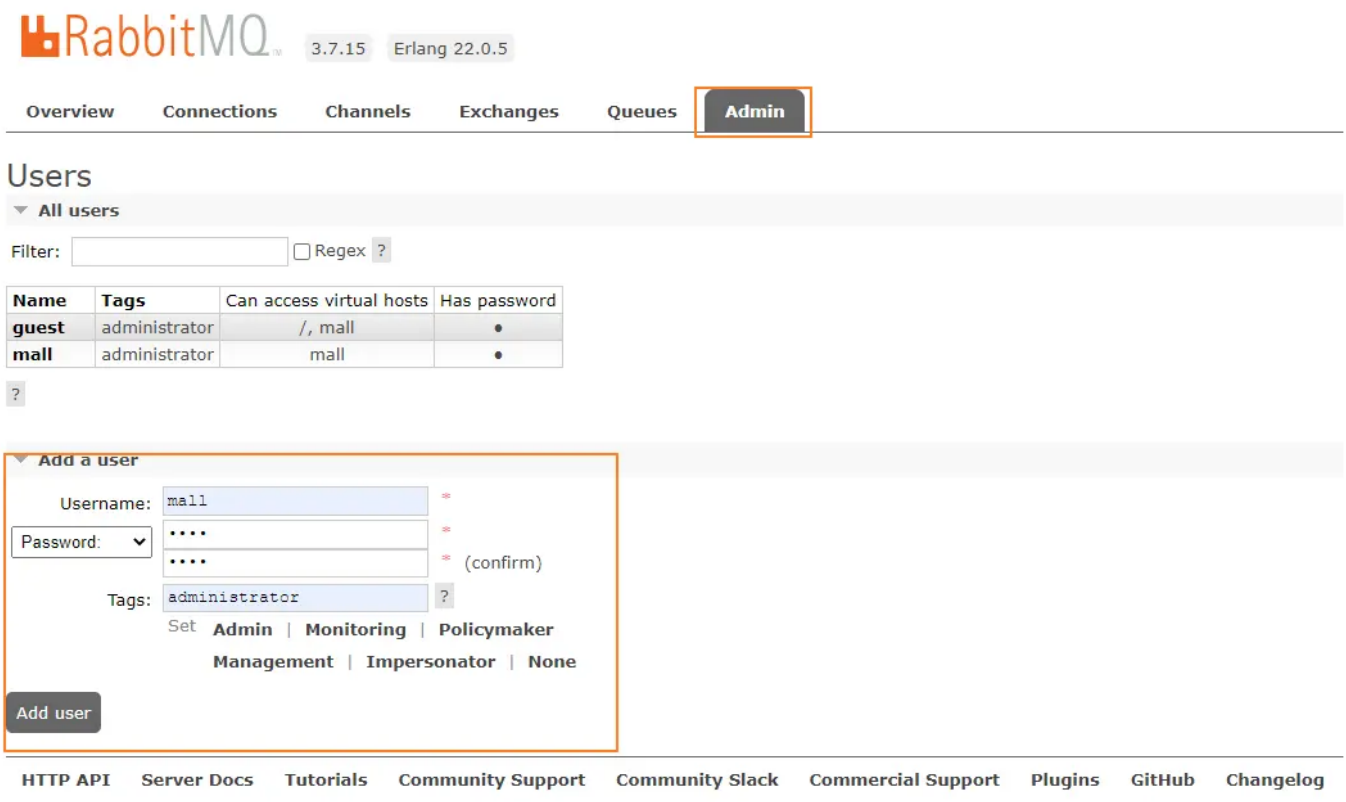
```
#登录容器的时候需要注意到容器支持的 shell 是什么。
kubectl exec -it <pod-name> -n <ns-name> bash
kubectl exec -it <pod-name> -n <ns-name> sh


root@cloud-rabbitmq-5b49d784c-gbr8m:/# rabbitmq-plugins enable rabbitmq_manage
Enabling plugins on node rabbit@cloud-rabbitmq-5b49d784c-gbr8m:
rabbitmq_management
The following plugins have been configured:
  rabbitmq_management
  rabbitmq_management_agent
  rabbitmq_web_dispatch
Applying plugin configuration to rabbit@cloud-rabbitmq-5b49d784c-gbr8m...
Plugin configuration unchanged.
```

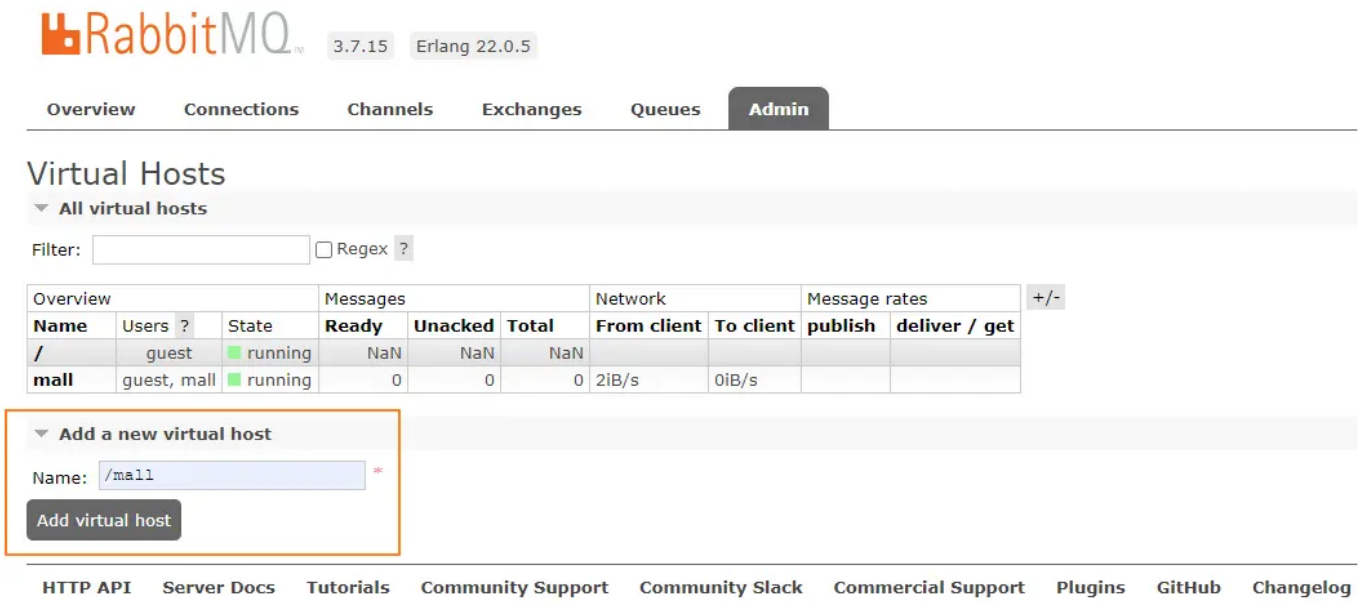
因为 RabbitMQ Service 使用 NodePort 的方式暴露控制台地址，比如 NodePort 为 15672。所以，第二步，我们访问地址 http:// 计算节点 IP:15672/ 地址，查看是否安装成功：

第三步，输入账号密码并登录 guest/guest。

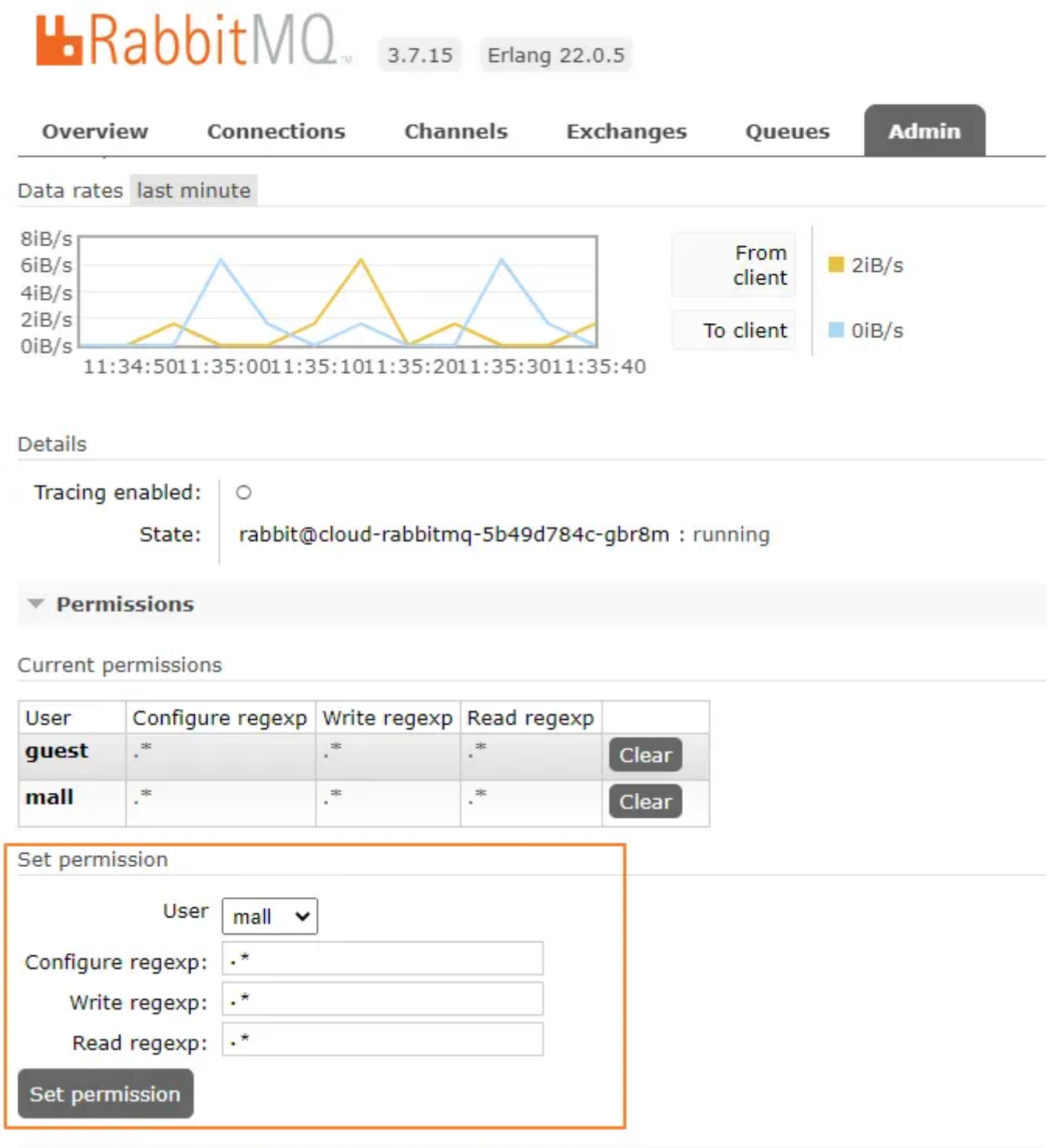
第四步，创建帐号并设置其角色为管理员 mall/mall。



第五步，创建一个新的虚拟 host 为 /mall。



第六步，点击 mall 用户进入用户配置页面，给 mall 用户配置该虚拟 host 的权限。



到这里，RabbitMQ 的初始化就完成了。

6.2. Elasticsearch

安装中文分词器 IKAnalyzer，并重新启动：

复制代码

```
#此命令需要在容器中运行
elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-i
```

7. 使用 yaml 资源文件部署应用

将项目 document/k8s 目录下的 yaml 资源文件中的 Dokcer 镜像，修改为自己的 Tag 并上传到 k8s 集群中执行：

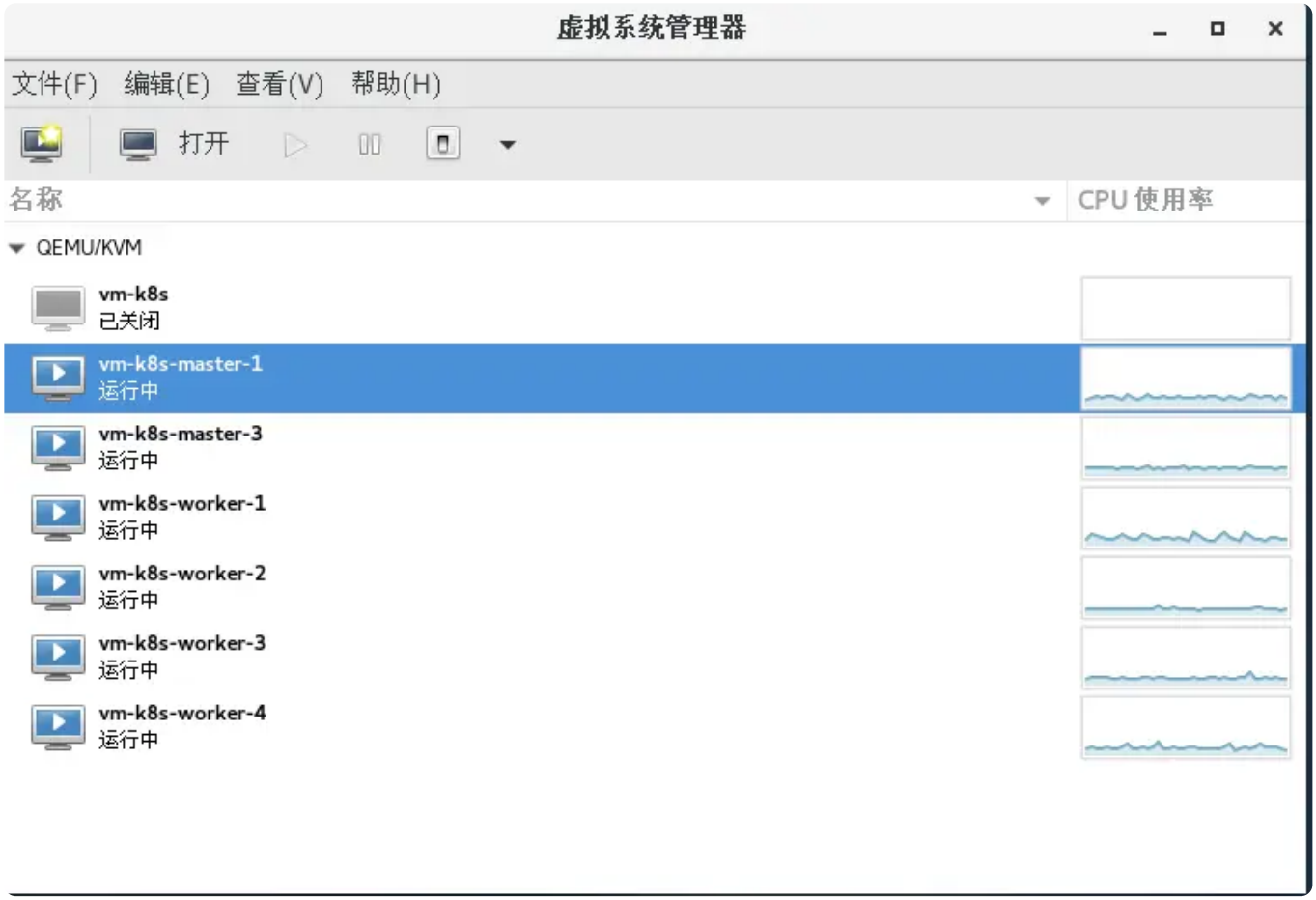
```
1 kubectl apply -f k8s/
```

复制代码

七. 运行效果展示

1. 服务器

2. 虚拟机



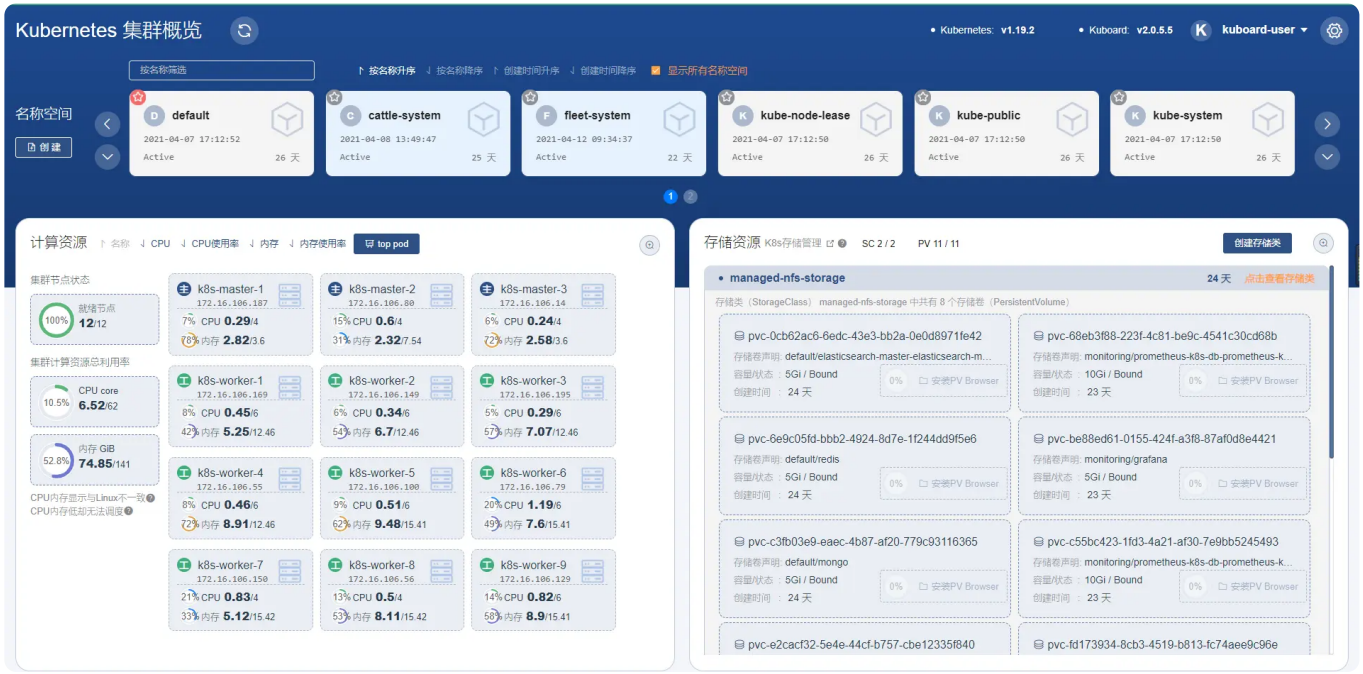
3. Kubernetes 集群

Kubernetes 集群：

复制代码

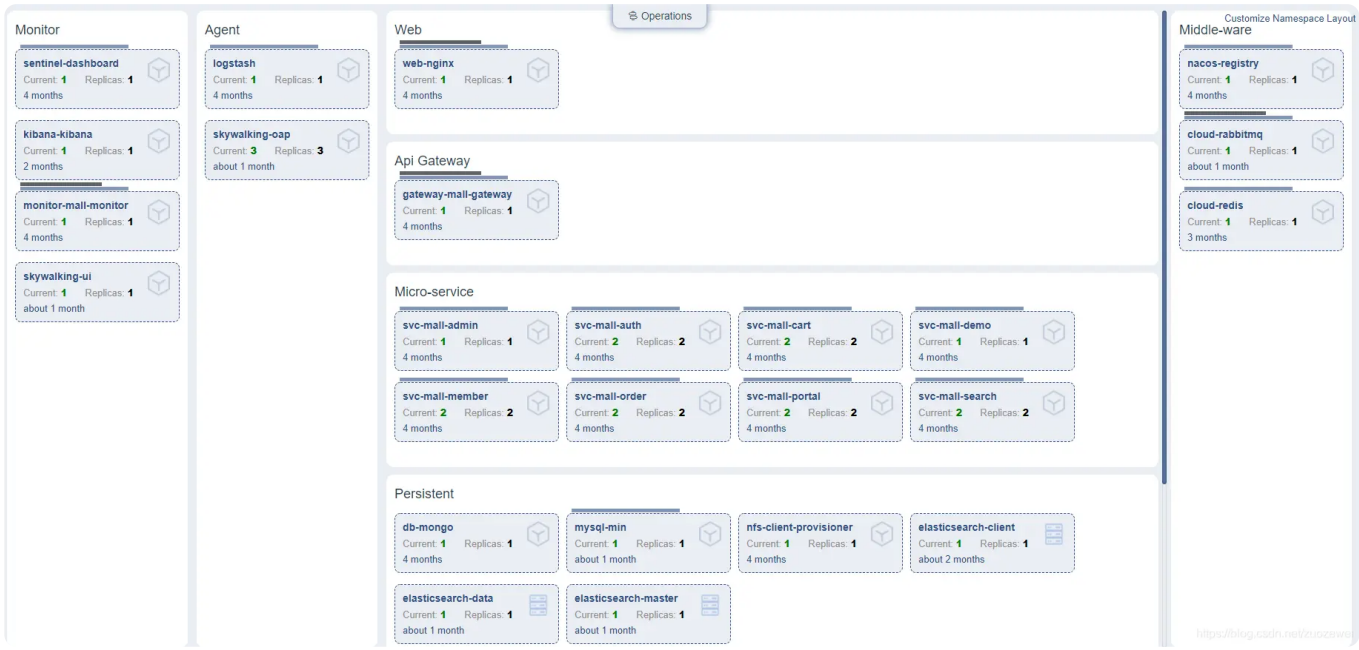
```
[root@k8s-master-1 ~]# kubectl get nodes
NAME           STATUS   ROLES    AGE   VERSION
k8s-master-1   Ready    master   26d   v1.19.2
k8s-master-2   Ready    master   26d   v1.19.2
k8s-master-3   Ready    master   26d   v1.19.2
k8s-worker-1   Ready    <none>   26d   v1.19.2
k8s-worker-2   Ready    <none>   26d   v1.19.2
k8s-worker-3   Ready    <none>   26d   v1.19.2
k8s-worker-4   Ready    <none>   26d   v1.19.2
k8s-worker-5   Ready    <none>   26d   v1.19.2
k8s-worker-6   Ready    <none>   26d   v1.19.2
k8s-worker-7   Ready    <none>   26d   v1.19.2
k8s-worker-8   Ready    <none>   26d   v1.19.2
k8s-worker-9   Ready    <none>   26d   v1.19.2
[root@k8s-master-1 ~]#
```

微服务管理：

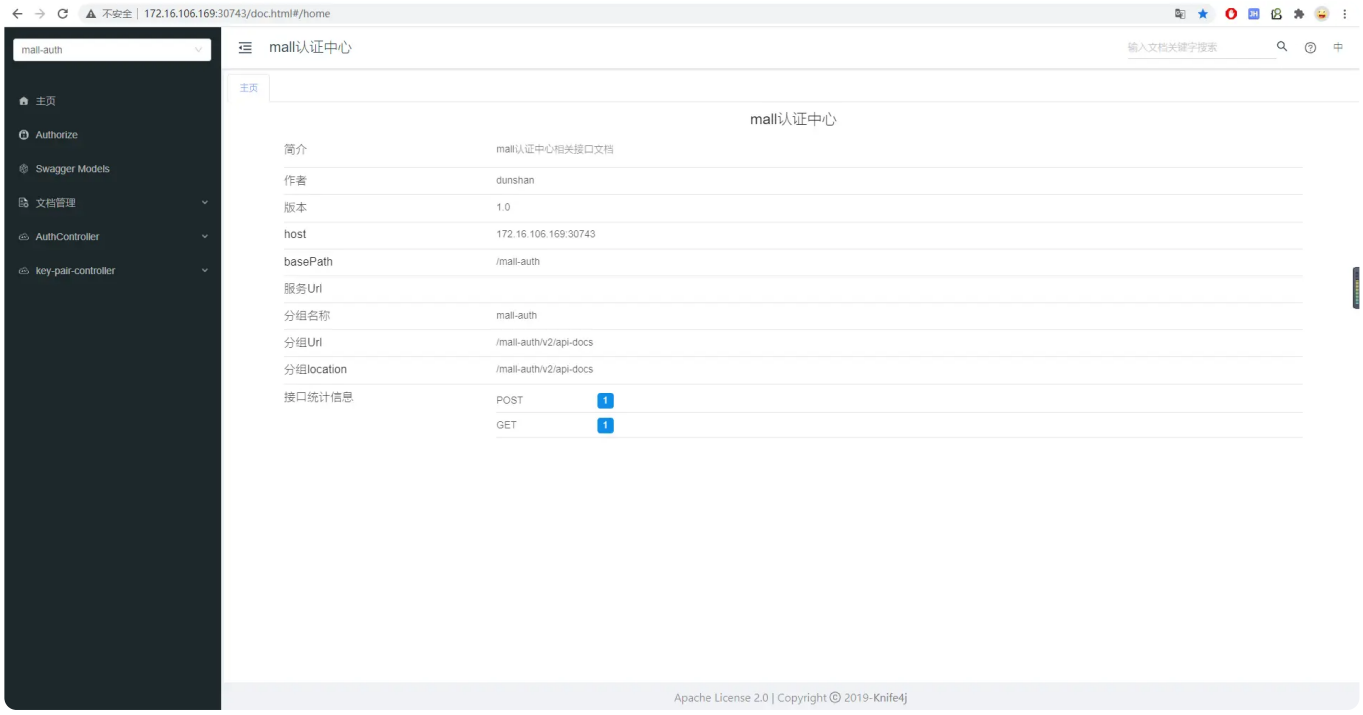


4. 微服务

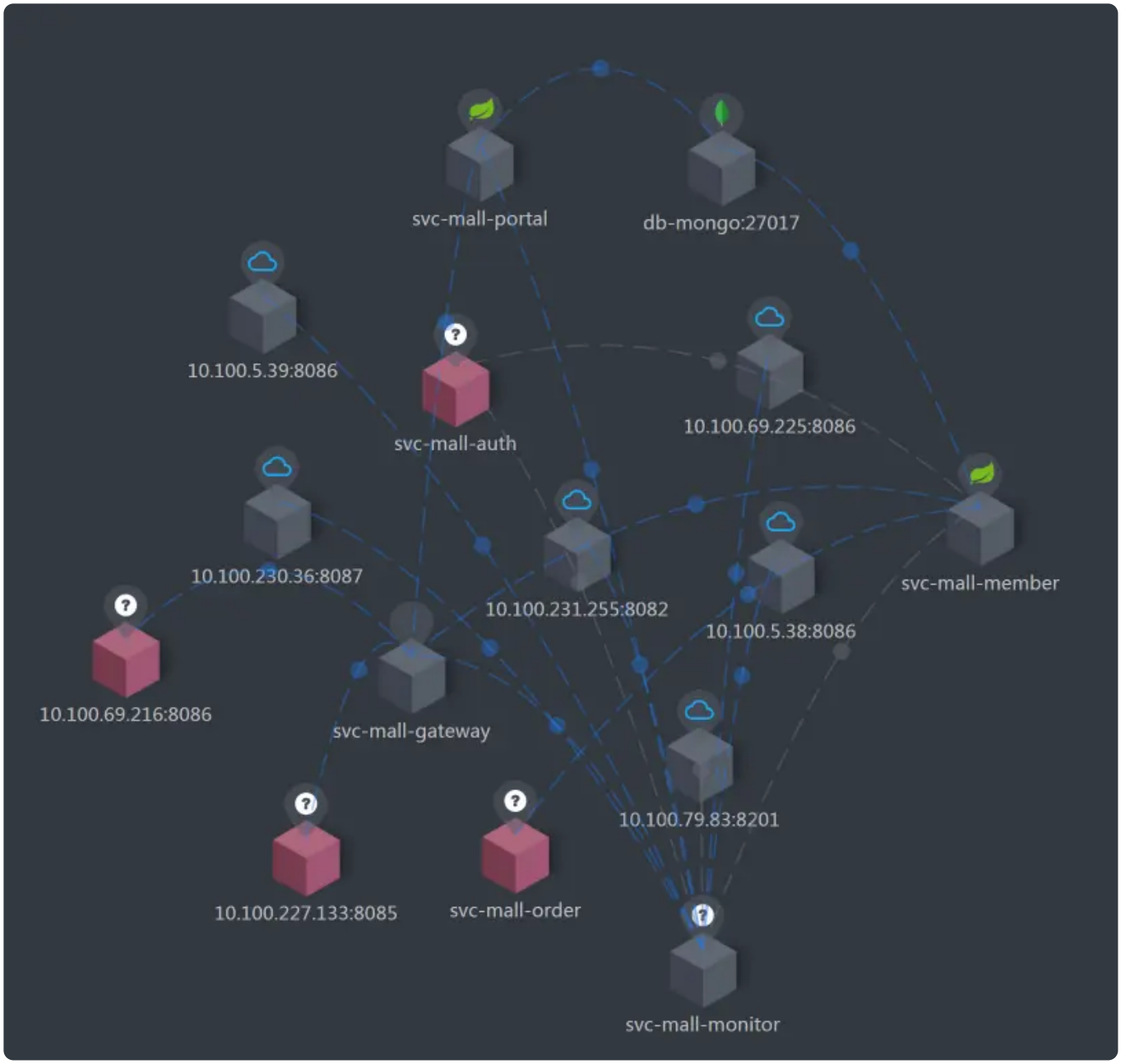
部署架构图：



API 文档：


不安全 | 172.16.106.169:30743/doc.html#/home
mall-auth
mall认证中心
输入文档关键字搜索
主页
Authorize
Swagger Models
文档管理
AuthController
key-pair-controller
主页
mall认证中心
简介 mall认证中心相关接口文档
作者 dunshan
版本 1.0
host 172.16.106.169:30743
basePath /mall-auth
服务Url
分组名称 mall-auth
分组Url /mall-auth/v2/api-docs
分组location /mall-auth/v2/api-docs
接口统计信息
POST 1
GET 1
Apache License 2.0 | Copyright © 2019-Knife4j

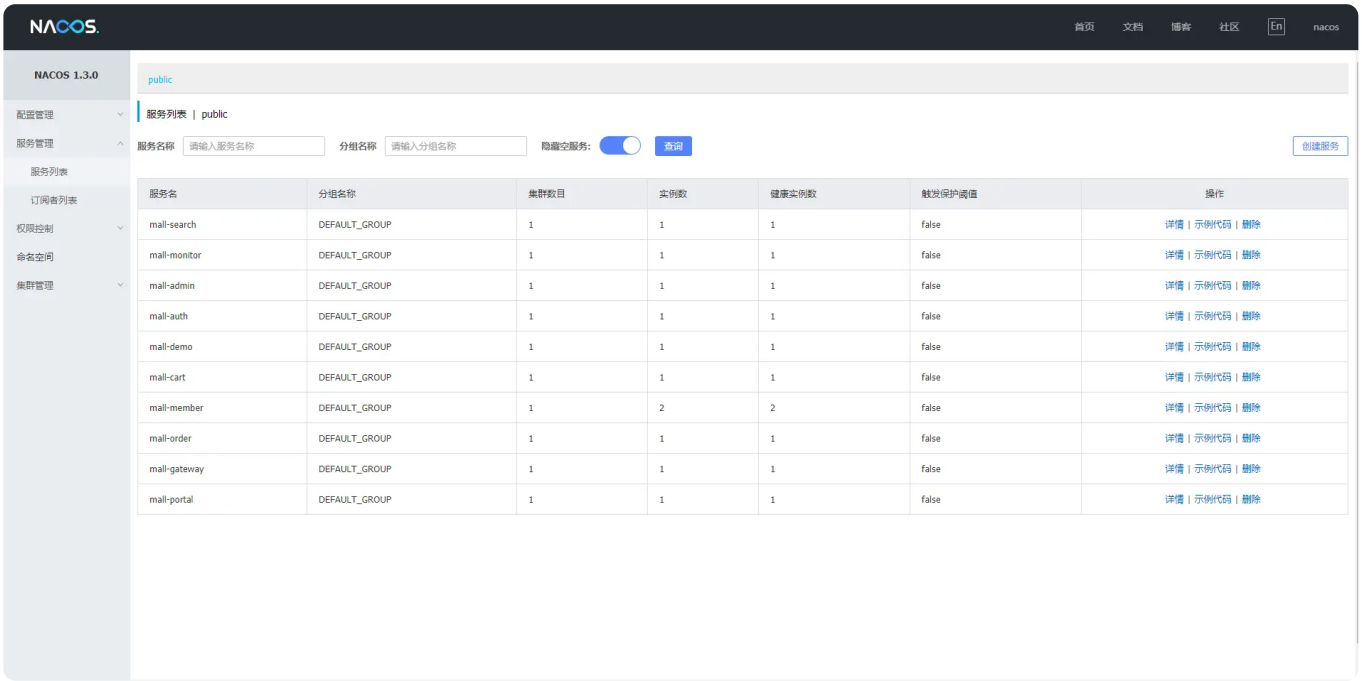
调用链监控：


svc-mall-portal
db-mongo:27017
10.100.5.39:8086
svc-mall-auth
10.100.69.225:8086
10.100.230.36:8087
10.100.231.255:8082
svc-mall-member
10.100.5.38:8086
10.100.69.216:8086
svc-mall-gateway
10.100.79.83:8201
10.100.227.133:8085
svc-mall-order
svc-mall-monitor

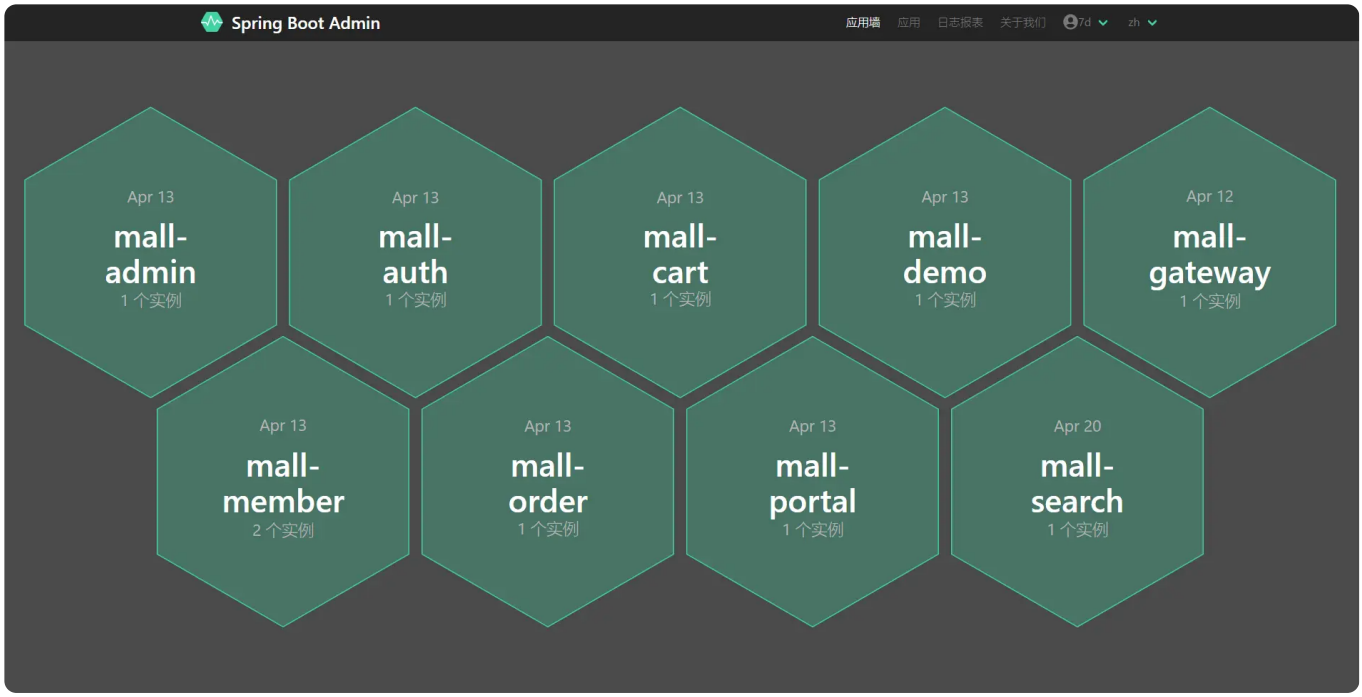
0
20
40
60
80
100
120
140
160
180
200
/mall-order/order/generateConfirmOrder
Http - spring-webflux
SpringCloudGateway/RoutingFilter
Unknown - spring-cloud-gateway
SpringCloudGateway/sendRequest
Unknown - spring-cloud-gateway
{POST}/order/generateConfirmOrder
Http - SpringMVC
Balancer/sso/feign/info
Http - Feign
/sso/feign/info
Http - OKHttp
{GET}/sso/feign/info
Http - SpringMVC
Lettuce/GET/onComplete
Cache - Lettuce
Lettuce/GET/accept
Cache - Lettuce
Lettuce/GET
Cache - Lettuce
Lettuce/GET/onComplete
Cache - Lettuce
Lettuce/GET/accept
Cache - Lettuce
Lettuce/GET
Cache - Lettuce
Mysql/JDBI/PreparedStatement/execute
Database - mysql-connector-java
Balancer/cart/list/promotionNew/{memberI...
Http - Feign
/cart/list/promotionNew/1110922
Http - OKHttp
{POST}/cart/list/promotionNew/{memberId}
Http - SpringMVC
Mysql/JDBI/PreparedStatement/execute
Database - mysql-connector-java
Balancer/promotion/cart/CartPromotion
Http - Feign
/promotion/cart/CartPromotion
Http - OKHttp
{POST}/promotion/cart/CartPromotion
Http - SpringMVC
Mysql/JDBI/PreparedStatement/execute
Database - mysql-connector-java
Balancer/member/coupon/cartList
Http - Feign
/member/coupon/cartList
Http - OKHttp
{POST}/member/coupon/cartList
Http - SpringMVC
Lettuce/GET/onComplete
Cache - Lettuce
Lettuce/GET/accept
Cache - Lettuce
Lettuce/GET
Cache - Lettuce
Mysql/JDBI/PreparedStatement/execute
Database - mysql-connector-java
Mysql/JDBI/PreparedStatement/execute
Database - mysql-connector-java

图片已做防盗链处理
请在原文件中访问该图片

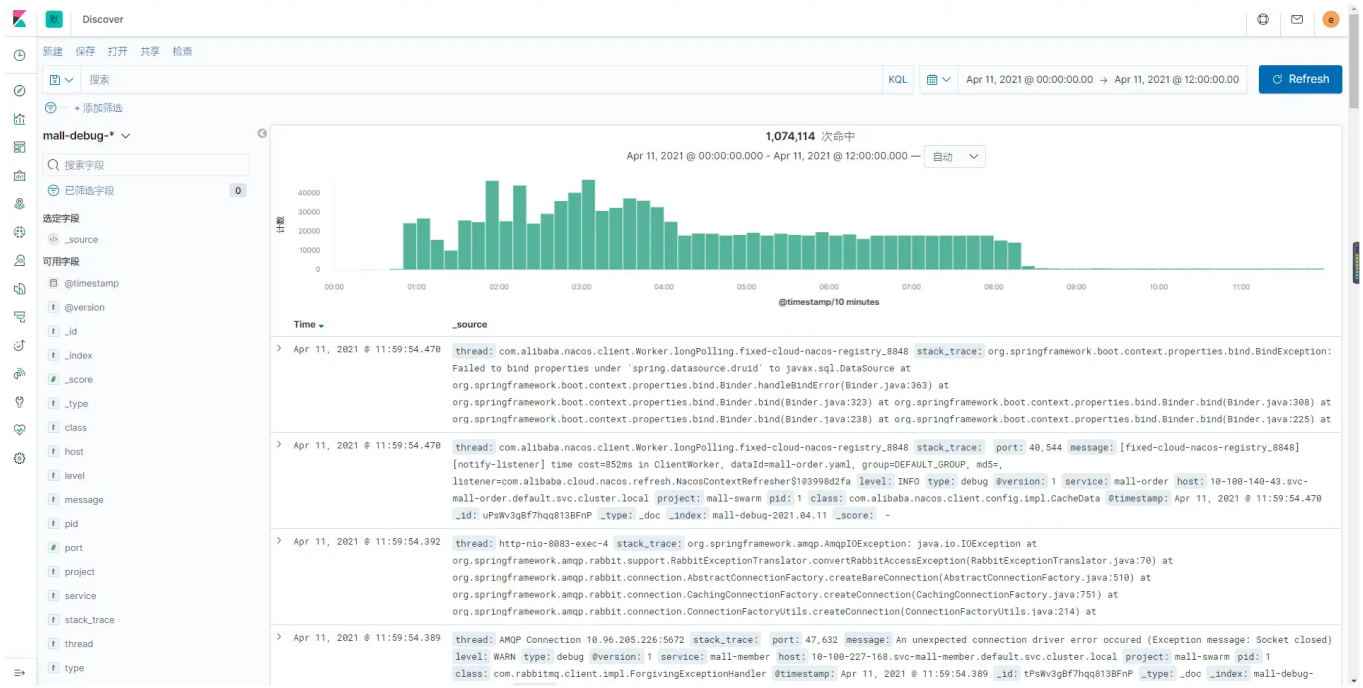
服务注册：


NACOS
首页 文档 博客 社区 En nacos
NACOS 1.3.0
配置管理
服务管理
服务列表
订阅者列表
权限控制
命名空间
集群管理
public
服务列表 | public
服务名称 请输入服务名称
分组名称 请输入分组名称
隐藏空服务:
查询
创建服务
服务名 分组名称 集群数目 实例数 健康实例数 触发保护阈值 操作
mall-search DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-monitor DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-admin DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-auth DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-demo DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-cart DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-member DEFAULT_GROUP 1 2 2 false 详情 | 示例代码 | 删除
mall-order DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-gateway DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除
mall-portal DEFAULT_GROUP 1 1 1 false 详情 | 示例代码 | 删除

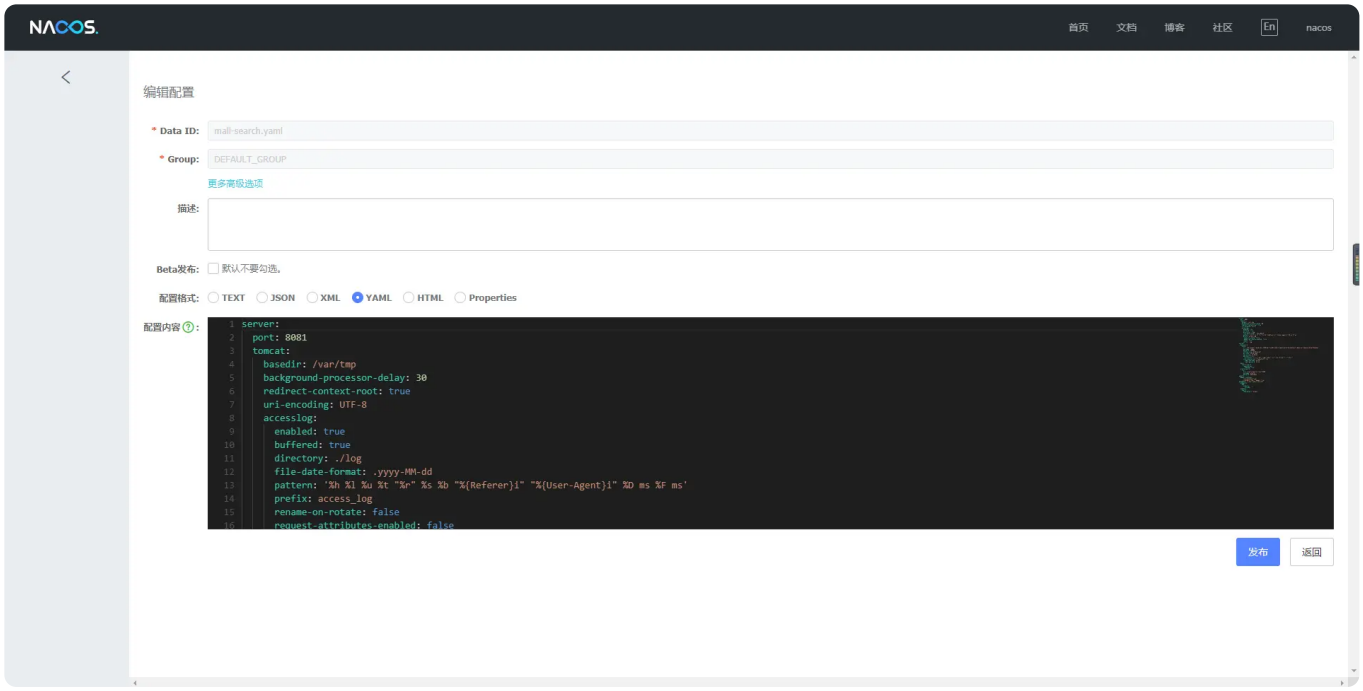
服务监控:


Spring Boot Admin
应用墙
应用
日志报表
关于我们
Apr 13
mall-admin
1 个实例
Apr 13
mall-auth
1 个实例
Apr 13
mall-cart
1 个实例
Apr 13
mall-demo
1 个实例
Apr 12
mall-gateway
1 个实例
Apr 13
mall-member
2 个实例
Apr 13
mall-order
1 个实例
Apr 13
mall-portal
1 个实例
Apr 20
mall-search
1 个实例

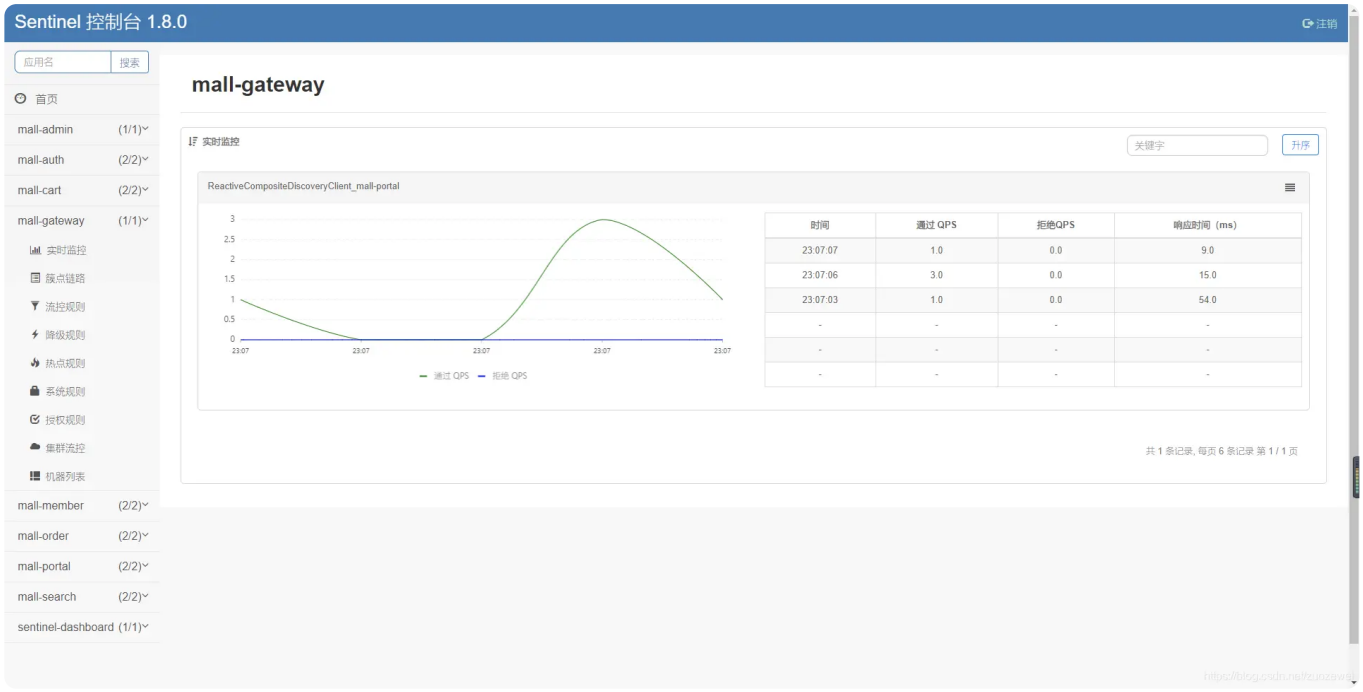
日志聚合：


Discover
Refresh
mall-debug-*
1,074,114 次命中
Time
_source

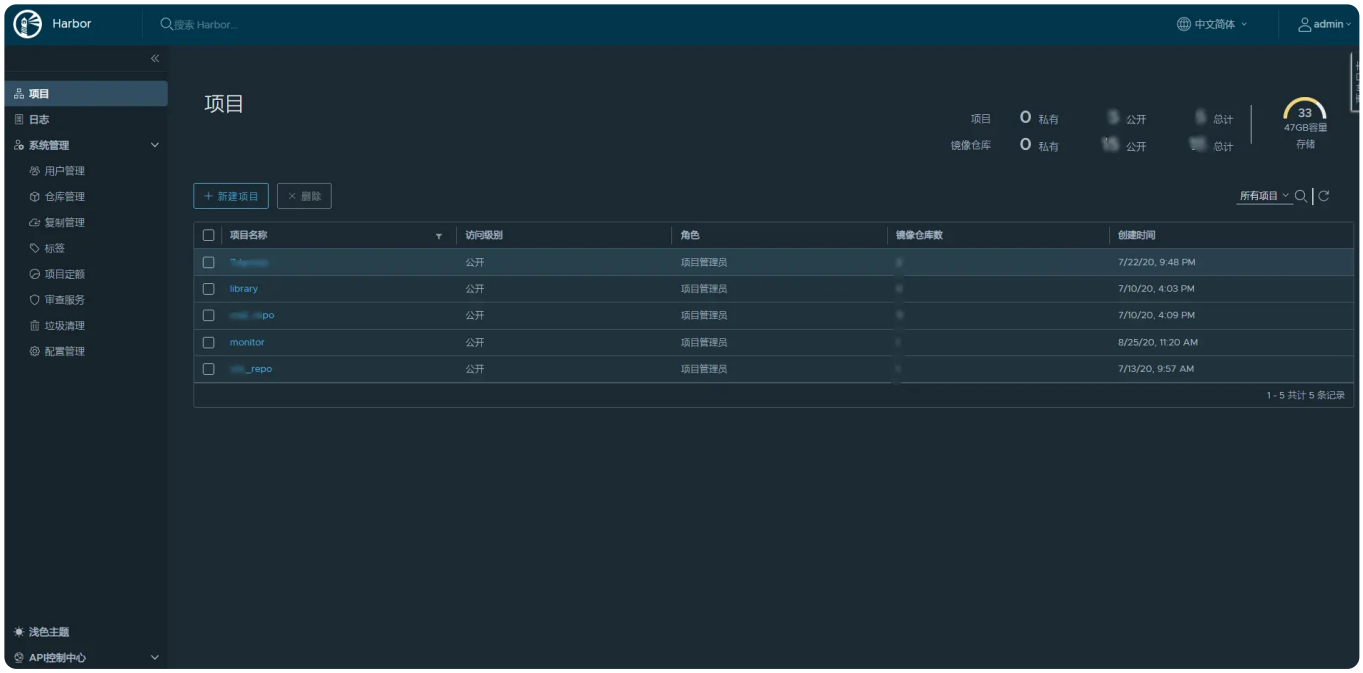
配置管理：


NACOS
首页
文档
博客
社区
En
nacos
编辑配置
* Data ID: mall-search.yaml
* Group: DEFAULT_GROUP
更多高级选项
描述:
Beta发布: 默认不要勾选。
配置格式: TEXT JSON XML YAML HTML Properties
配置内容:
server:
port: 8081
tomcat:
basedir: /var/tmp
background-processor-delay: 30
redirect-context-root: true
uri-encoding: UTF-8
accesslog:
enabled: true
buffered: true
directory: ./log
file-date-format: .yyyy-MM-dd
pattern: '%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-Agent}i" %D ms %F ms'
prefix: access_log
rename-on-rotate: false
request-attributes-enabled: false
发布
返回

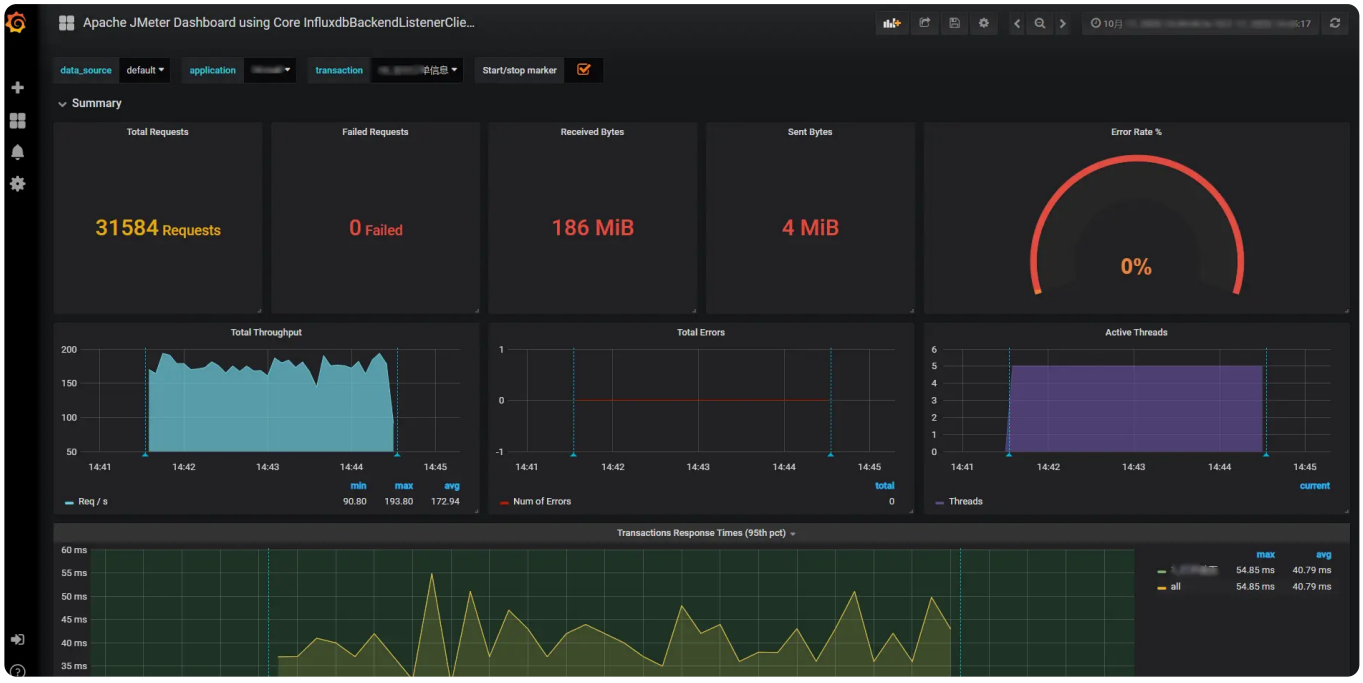
系统保护：


Sentinel 控制台 1.8.0
注销
应用名
搜索
首页
mall-admin (1/1)
mall-auth (2/2)
mall-cart (2/2)
mall-gateway (1/1)
实时监控
簇点链路
流控规则
降级规则
热点规则
系统规则
授权规则
集群流控
机器列表
mall-member (2/2)
mall-order (2/2)
mall-portal (2/2)
mall-search (2/2)
sentinel-dashboard (1/1)
mall-gateway
实时监控
关键字
升序
ReactiveCompositeDiscoveryClient_mall-portal
时间
通过 QPS
拒绝QPS
响应时间（ms）
23:07:07 1.0 0.0 9.0
23:07:06 3.0 0.0 15.0
23:07:03 1.0 0.0 54.0
通过 QPS
拒绝 QPS
共 1 条记录，每页 6 条记录 第 1 / 1 页

容器仓库：



压力引擎：



5. 资源监控

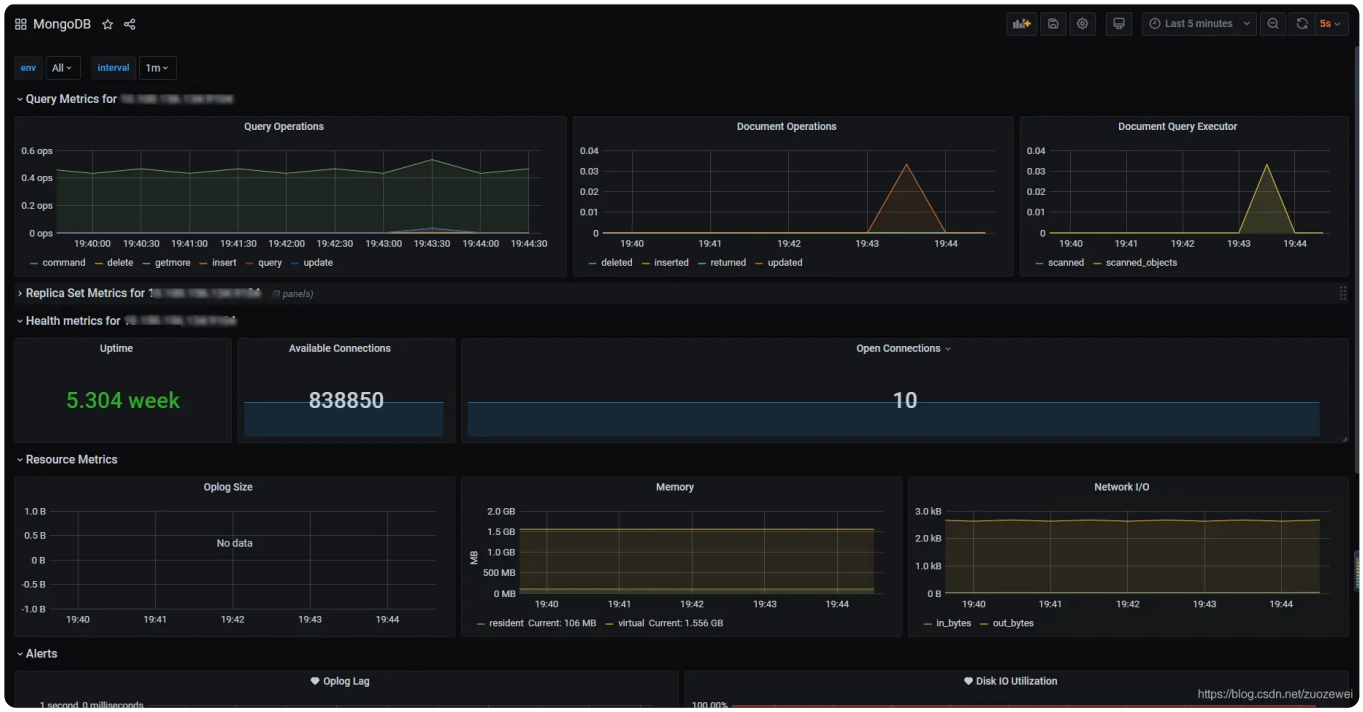
Kubernetes 集群资源监控：

Linux 资源监控：

MySQL 资源监控：

RabbitMQ 资源监控：

MongoDB 数据库资源监控：

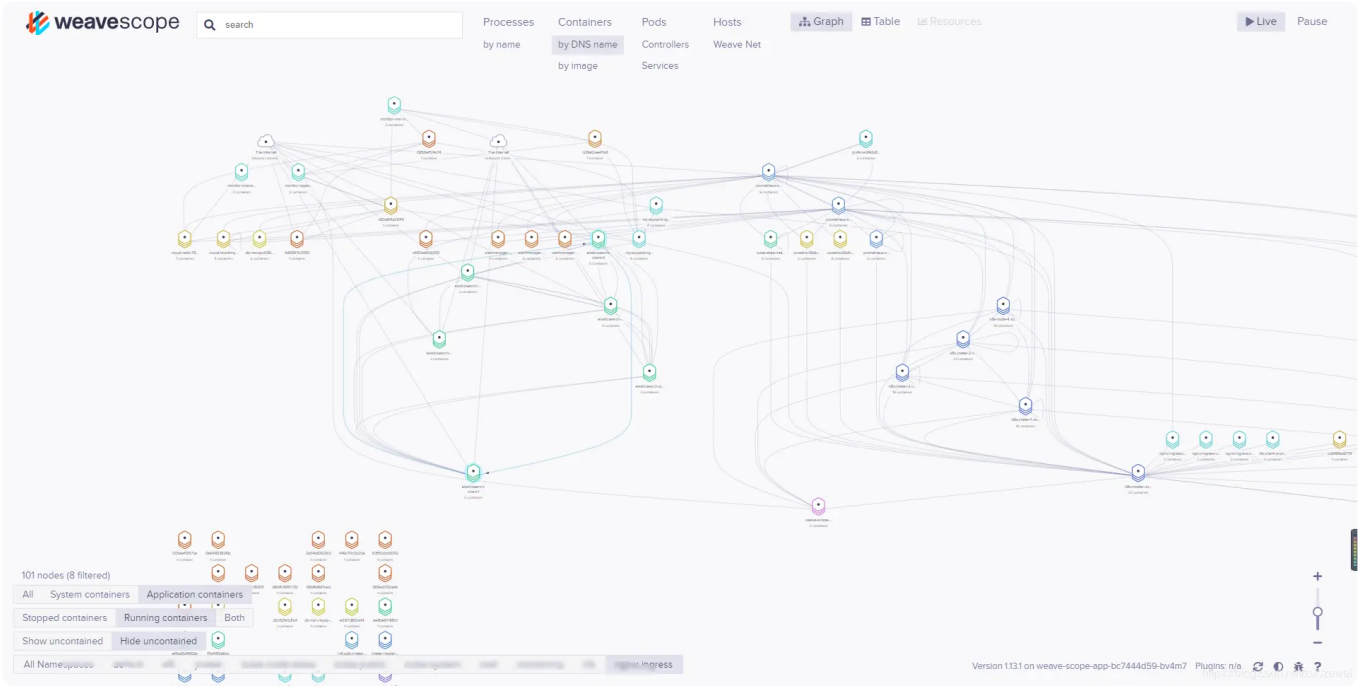


Kubernetes etcd 资源监控：

Kubernetes API Server 资源监控：

Default / Kubernetes / API server
The SLO (service level objective) and other metrics displayed on this dashboard are for informational purposes only.
Availability (30d) > 99.000%
99.735%
ErrorBudget (30d) > 99.000%
Read Availability (30d)
99.708%
Read SLI - Requests
Read SLI - Errors
Read SLI - Duration
Write Availability (30d)
99.804%
Write SLI - Requests
Write SLI - Errors
Write SLI - Duration
Work Queue Add Rate
Work Queue Depth

Kubernetes 服务拓扑：



八. 总结

这节课的内容包括了物理环境的说明、技术组件的具体搭建过程、示例系统的搭建过程以及运行效果。经过上面所有的步骤，我们就把整个课程涉及的所有技术组件、示例系统完全搭建起来了。

而我之所以选择这样的技术栈，主要有三方面的考虑：

1. 核心优势

任务调度：为集群系统中的任务提供调度服务，自动将服务按资源需求分配到资源限制的计算节点；

资源隔离：为产品提供管控与服务节点隔离能力，保证研发应用和管控服务不产生相互的影响；

高可用能力：自动监控服务运行，根据运行情况对失效的服务进行自动重启恢复；

网络互联互通能力：提供统一的 IP 地址分配和网络互通能力；

统一编排管理能力：结合 Gitlab 和 k8s ，对输出的产品进行统一的编排管理；

公共产品组件可以为团队提供统一部署、验证、授权、调度和管控能力，为私有云服务提供基础性的支撑。

2. 核心设施平台（IaaS 云）

提供计算、网络、存储等核心资源设备的虚拟化；

支持不同操作系统，包括主流的 Win 和 Linux 系统；

提供主要的三种服务：云主机、云网络、云硬盘；

提供可视化 Web UI；

提供 k8s 集群（容器云）规划、部署和运营；

支持多种计算、存储和网络方案。

3. 基础服务平台（PaaS 云）

提供数据存储：支持常见 NFS 、Ceph RBD、Local Volume 等；

提供应用服务：支持自愈和自动伸缩、调度和发布、负载均衡等；

提供运维管理：支持日志监控、资源监控、消息告警等。

我们这个系统采用的技术栈，是当前技术市场中流行的主流技术栈，这样的环境具有很高的借鉴价值。而且，从我们要表达的 RESAR 性能分析架构和逻辑来说，也说明 RESAR 性能分析理念是足以支撑当前的技术栈的。

参考资料汇总

1. **CentOS 7 的部署**：HP 服务器安装 CentOS 7

2. **KVM 的优化重点**：关于 KVM 虚拟化注意的二三事整理

3. **KVM 的安装和使用**：Linux KVM 安装使用手册

4. **Kubernetes 集群搭建**：

   单 Master 集群：使用 kubeadm 安装单 master kubernetes 集群（脚本版）

   高可用方案： Kubernetes 高可用集群落地二三事

5. **Kubernetes 的使用基础**：

Kubernetes 集群基本概念

k8s 入门篇 -Kubernetes 的基本概念和术语

K8s 命令篇 -Kubernetes 工作实用命令集结号

Kubernetes 集群常用操作总结

6. Kubernetes 网络插件选型：Kubernetes 网络插件（CNI）超过 10Gbit/s 的基准测试结果

7. NFS 部署： Kubernetes 集群部署 NFS 网络存储

8. Ceph 部署： Kubernetes 集群分布式存储插件 Rook Ceph 部署

9. Kubernetes 下的 MySQL 部署：如何在 Kubernetes 集群中搭建一个复杂的 MySQL 数据库

10. Elasticsearch 集群的部署：Kubernetes Helm3 部署 Elasticsearch & Kibana 7 集群

11. JMeter 的部署：

二进制：性能工具之 JMeter+InfluxDB+Grafana 打造压测可视化实时监控

Kubernetes：Kubernetes 下部署 Jmeter 集群

12. 镜像仓库 Harbor 的部署：Kubernetes 集群仓库 harbor Helm3 部署

13. Nacos 的部署：

Docker 单机模式： Nacos Docker 快速开始

Kubernetes：Kubernetes Nacos

14. Redis、RabbitMQ、MongoDB 单机部署的部署：Kubernetes 集群监控 kube-prometheus 自动发现

15. **Logstash 的部署**：整合 ELK 实现日志收集

16. **Kubernetes 集群资源监控的部署**：

- Kubernetes 集群监控 kube-prometheus 部署
- Kubernetes 集群监控 controller-manager & scheduler 组件
- Kubernetes 集群监控 ETCD 组件

17. **日志聚合部署的部署**：Kubernetes 集群日志监控 EFK 安装

18. **依赖组件的部署**：Kubernetes 集群监控 kube-prometheus 自动发现

19. **APM 链路跟踪的部署**：Kubernetes + Spring Cloud 集成链路追踪 SkyWalking

20. **微服务项目介绍**：《高楼的性能工程实战课》微服务电商项目技术全解析

21. **其他学习资料推荐**：

- SpringCloud 日志在压测中的二三事
- 高楼的性能工程实战课之脚本开发
- 《高楼的性能工程实战课》学习所推荐的知识点

分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 0 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 29 | 异常场景：如何模拟不同组件层级的异常？

下一篇 30 | 如何确定生产系统配置？

更多学习推荐

精选留言 (4)

写留言

雄鹰
2021-05-29

老师思路清晰，羡慕佩服，若能学会，也是大牛了

展开

作者回复: 想做到，并动手做，就会成为大牛。
所谓大牛就是吃的苦更多，消耗的时间更多。
世上哪有躺着成为大牛的，不过是逼着自己慢慢学习而已。哈😊

4

微思
2021-06-11

赞！！！

展开

作者回复: 精辟！哈哈

WBF

2021-05-29

搭建整套环境，需要详细阅读并实践这些参考资料不下于20篇，知识点涵盖全面，实用性很强，就是搭建成本有点高哈，感谢高老师的良苦用心，收获很大

展开

作者回复: 学习嘛，总是要付出点成本的。你可以在低成本环境中试一下能不能玩起来。哈😊

道长

2021-05-28

感觉能把这些搞懂，就很厉害了

展开

作者回复: 所以呀，赶紧厉害起来吧。动手是唯一的成长路径。